A Combined Potential Function and Graph Search Approach for Free Gait Generation of Quadruped Robots

Yam Geva and Amir Shapiro

***Abstract*— This paper presents an algorithm for planning the foothold positions of quadruped robots on irregular terrain. The input to the algorithm is the robot kinematics, the terrain geometry, a required motion path, as well as initial posture. Our goal is to develop general algorithm that navigate quadruped robots quasi-statically over rough terrain, using an APF (Artificial Potential Field) and graph searching. The algorithm is planning a sequence set of footholds that navigates the robot along the required path with controllable motion characteristics. Simulations results demonstrate the algorithm in a planner environment.**

I. Introduction

In conventional motion planning, a wheeled mobile robot navigates toward a goal configuration while avoiding collision with obstacles. However, many motion planning problems are more suited for legged robots that interact with the environment in order to achieve stable locomotion. Gaits can be classified in several ways, The common classification is according to the locomotion type - dynamic or quasi-static. In quasi-static walking the body is at equilibrium while one leg is moving and the inertial effects due to moving parts of the robot are kept small relative to forces and torques of interaction between the robot and the environment. In dynamic walking, the robot system has a stable limit cycle and the locomotion algorithm is taking care of the inertial and external forces. Another gait classification is according to the types of gait pattern generation - periodic and non-periodic. in periodic gaits there is a fixed pattern of legs movement sequence and foothold selection. Walking machines with periodic gaits can move with large stability margin and can be easily controlled. The disadvantage of a fixed gait pattern is that it can be used only on relatively flat terrain. In a non-periodic gaits, also called free gaits, there is no pre-planned sequence of leg lifting and foothold placement and the movement is controlled by a set of rules, this method increases the terrain adaptability of free gaits walking machines. There are two main methods used for free gaits generation: search-based and rule-based. The ruled-based method plans the legs sequence and placement by the use of a set of rules designed by the programmer [1], [6], [7], [8], learned automatically [12], or derived from biological mechanisms found in nature [4]. In the search-based method, a different series of possible robot motions is generated and a search for a valid motion or motion sequences is applied. The most significant drawbacks of the search method is the high computations requirements for searching a very large number of possible motion series, especially when searching for the optimal path [17] and the need of a prior knowledge about the terrain characteristics. However, we believe that the computers technology for high complexity calculations and terrain data gathering sensors has come to an adequate level for the use of a search-based method and the use of such technologies can significantly improve results for gait generation over rough terrains. Due to this reason, the algorithm presented in this paper uses a graph search-method combined with an artificial potential field and a post-processing stage in order to accomplish the path generation and locomotion task. The graph search method for legged locomotion has been studied before first by Pal and Jayarajan [15], who used the heuristic graph search method A* to generate a free gait along a straight-line for a quadruped and later extended to an omnidirectional gait [14], [5]. Using graph search for free gait generation without heuristics in a restricted environment has been studied in [2]. The suggested algorithm introduce a new method for free-gait generation and some significant improvements for prior work. The use of APF as a tool for solving navigation problems is known and well studied, the use of APF for gait generation has also been done in [9], [10], which used it to define and avoid obstacles. We use the APF in graph search in order navigating along a required path. We address the APF local minima problem using graph search.

The paper is organized as follows: Section II present the posture and stability definitions and parameters used by the algorithm. Section III present the two stages of the algorithm itself with details and discuss real-time applications. Section IV presents simulations results. Finally, conclusions and future work are presented.

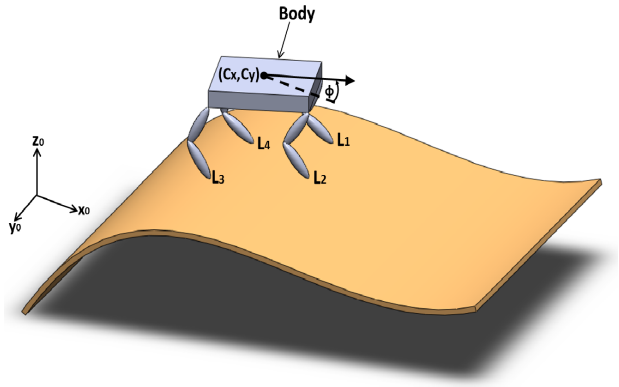
II. Postures And Stability

The suggested algorithm is suitable for any four-legged walking machine with three degrees of freedom in each leg and assumes the robot kinematics is known. We also assume that the legs movement affect the robot center of mass is minor. As shown in Figure 1(a), a quadruped posture can be defined by a set of parameters, we define a specific posture by the vector $q \in \mathbb{R}^{11}$:

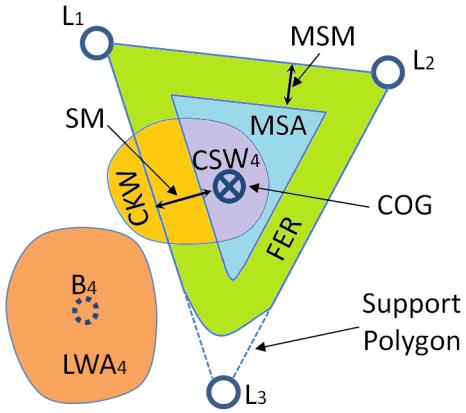
$$q = (COG, L)^T, \tag{1}$$

the $COG = (C_x, C_y, \phi)$ vector defines the center of gravity location and the body yaw angle (Figure 1(a)). The $L = (L_1, L_2, L_3, L_4)$ vector defines the footholds of all four legs

This work was partially supported by the Israeli Ministry of Defence.
Y. Geva and A. Shapiro are with the Department of Mechanical Engineering, Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105,Israel [gevay,ashapiro]@bgu.ac.il



(a) Posture parameters



(b) Top view of three footholds posture for $ML = 4$ (fourth leg is the Moving Leg)

Fig. 1. Postures (a) and stability (b) definitions

where L_i=(L_{i_x}, L_{i_y}) is the coordinates of leg i foothold. The vector $B = (B_1, B_2, B_3, B_4)$ defines the positions of the legs scapula joints, B_i=(B_{i_x}, B_{i_x}) is the coordinates of leg i scapula joint. We also assume a constant height above the ground and that the body is always in parallel with the plane which defined by the current three supporting legs footholds.

A stable posture is defined as a posture in which the center of gravity is above the edge of the FER (Feasible Equilibrium Region) as described in [13], *this is not necessarily the conventional supporting polygon of the three legs footholds*, especially on non planar ground. This stability definition is suitable to planar and irregular terrains where friction effects are significant.

As shown in Figure 1(b) ,we define the stability margin, *SM*, to be the the minimal allowed distance from the center of gravity to the FER edge, the *SM* value has length units and can be used as a tool that allows us to quantitatively assess the current posture stability. In our research we consider only quasi-static locomotion so we need to ensure stability while standing on three legs assuming that when the fourth leg will touch the ground, the robot remains in equilibrium. We define the $MSA_i \in \mathbb{R}^2$ (Margin of Stability Area) as the set of all points inside the FER which has a margin of stability

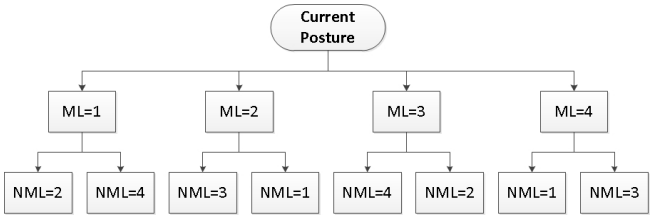


Fig. 2. Eight possible double step sequences for each DFS iteration

larger than the *MSM* (Minimum *SM*) during the movement of leg i. We define *ML* as the index (1,2,3,4) of the current Moving Leg, and $LWV_i \in \mathbb{R}^3$ (Leg Workspace Volume) as the volume in which the leg i footpad can move to while the body is at a pre-defined desired *MSM*. This volume is a function of the leg kinematics and the location of B_i. Between two consecutive leg lifting maneuvers there exist a four legged posture, during which the *COG* can be moved. Hence, in order to calculate the LWV_{ML}. Denote $CKW \in \mathbb{R}^3$ (*COG* Kinematic Workspace) as the area in which the *COG* can kinematically move to while standing on all four legs:

$$CKW = f_{CKW}(L), \tag{2}$$

where f_{CKW} is the inverse kinematics function of the robot, it computes the feasible *COG* locations for a given L.
The $CSW_i \in \mathbb{R}^3$ (*COG* Stable Workspace) is defined as the intersection of the *CKW* and the MSA_i:

$$CSW_i = MSA_i \cap CKW. \tag{3}$$

Therefore in order to meet the kinematic reachability constraint and the equilibrium (stability) constraints the *COG* must be within the CSW_i. A specific *COG* from the *CSW* determines the location of B_i and therefore defining the LWV_i. Hence, for a given posture q and a specific *ML*, we get:

$$LWV_{ML} = f_{LWV}(COG), \tag{4}$$

where f_{LWV} is the inverse kinematics function which computes the feasible footpad locations of leg *i* for a given B_i. The $LWA_i \in \mathbb{R}^2$ (Leg Workspace Area) is the intersection of the LWV_i and the ground surface, each point in this area defines a kinematically feasible and stable foothold position. It is possible that the *LWA* will be an empty set. This happens when there is no intersection of the *LWV* with the ground.

Moving from one posture to another can be done continuously or discontinuously, the difference is the location of the *COG* while the *ML* is in transition phase. In continuous waking [6], the COG is moving together with the *ML* although it has to remain inside the FER in order to maintain equilibrium. In discontinuous walking [16], [7], the *COG* is static while the *ML* is in transition phase and one step can be divided into two segments, the *COG* motion and the ML motion. The algorithm presented here is based on a combination of these two walking types. The algorithm searches for a sequence of two legs movement that will sequently move while the *COG* remains at the same location. However, the ϕ angle can be changed before the second step

is being conducted. This type of step sequence is different from the type described in prior works [11], [16] in which the location of the two footholds are known while planning the location of the *COG* and the location will be a result of the foothold selection algorithm.We define the *NML* (Next Moving Leg) to be the index (1,2,3,4) of the second leg in the sequence. Calculation of the CSW_{NML} and therefore the calculation of the LWA_{NML} depends on the L_{ML} because only after fixing the L_{ML} (the first leg in the sequence) we can calculate the *CKW* of the intermediate posture (after the first step in the sequence) using the f_{CKW} function.

III. Footholds Selection Algorithm

The suggested algorithm includes a main processing stage and a post-processing stage, the first stage include the graph building and searching, we use the DFS (Depth First Search) algorithm in order to navigate the *BP* (Body Position) along the path. The BP is a function of the robot posture while standing on all four legs:

$$BP = f_{BP}(q). \tag{5}$$

Because of the COG three degrees of freedom while standing on all four legs, this function is needed to be defined and be used as one of the algorithm inputs. We can defined the f_{BP} function to be the average coordinates of all four footholds:

$$f_{BP} = (\frac{L_{1_x}+L_{2_x}+L_{3_x}+L_{4_x}}{4}, \frac{L_{1_y}+L_{2_y}+L_{3_y}+L_{4_y}}{4}), \tag{6}$$

but it can also be defined differently. This is not the position of the *COG*. In order to build the graph we need to define the graph nodes and edges. A node on the graph represent a specific four legs equilibrium posture and an edge is a possible step sequence to be conducted between two consecutive four legs postures as described in section II.

In order o find a path from start configuration to target we search the graph for possible path using the DFS (Depth First Search) algorithm. In each iteration of the DFS algorithm, the nodes need to be graded, we use an APF (Artificial Potential Field) in order to perform this task. The potential function is a normalized weighted average of three components:

$$U = \frac{w_1 \cdot U_{Dist1} + w_2 \cdot U_{Dist2} + w_3 \cdot U_{ang}}{w_1 + w_2 + w_3}, \tag{7}$$

Where U_{Dist1} is the normalized distance along the path of the BP from the end of the path, U_{Dist2} is the normalized distance of the *BP* from the path, U_{ang} is the normalized absolute value of the difference between the body heading and the path direction, the body heading is calculated by averaging the angles of the two side edges of the four legs footholds polygon. w_1, w_2 and w_3 denote weights for each component. By adjusting the weights, it is possible to control the motion characteristics, for example, by decreasing w_3 to zero, the robot will move along the path but will not necessarily face towards the target point.

The algorithm searches for all possible step sequences, Figure 2 shows the eight possible sequences. Those possible sequences are the possibilities for *NML* for each *ML* while moving only the two adjacent legs because it is impossible to locate the *COG* in a position that will ensure equilibrium for two diagonal pair of legs and sequens of the same leg cannot be better then a single step with the specific leg. For each sequence j (j=1...8) we need to solve the following minimization problem:

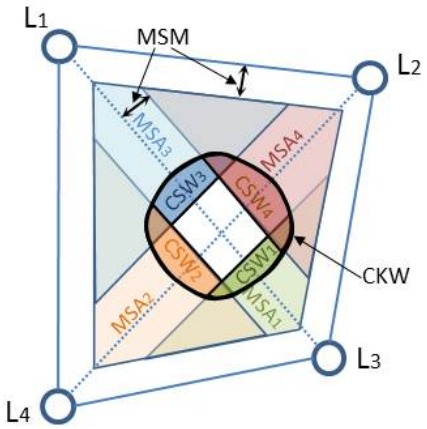


Fig. 3. Constraint C requirements

$$\begin{aligned} &q_j = argmin(U_j), \text{ subject to:} \\ &COG = (C_x, C_y, \phi) \in CSW_{ML_j}, \\ &L_{ML_j} = (L_{ML_x}, L_{ML_y}) \in LWA_{ML_j}, \\ &L_{NML_j} = (L_{NML_x}, L_{NML_y}) \in LWA_{NML_j}. \end{aligned} \tag{8}$$

where $U_j \in \mathbb{R}^7$ is the potential function value as defined in equation (7). By solving this problem for all eight sequences and choosing the sequence with $U_{min} = min(U_j, j = 1...8)$ we get the step sequence L_{ML} and L_{NML} footholds and *COG* that will reduce the potential function as much as possible for the given initial posture. The algorithm should theoretically run until the U value reach zero but this is not likely to happen because of system inaccuracies such as actuators imprecision, or inaccurate ground model, so we define a Low Potential Threshold (*LPT*) value as the required potential goal. This method of sequences search can also produce a single step with no sequel step because a single step is equal to a two steps sequence where L_{ML} remain at the same position.

The use of the DFS algorithm allows us to set a specific depth of search. This depth is the number of steps to be planned in advance. Setting this depth to a high value will lead to a well planned path because the algorithm will be able to avoid obstacles before reaching them. The cost of this deep search is an high computation time. When the depth is not covering the final target local minima can be found while searching for the path.

In order to avoid deadlock or local minima situations in which there is no possible step sequences of any type and to enhance the omnidirectional capabilities we apply constraints on the *LWA* as follows:

Constraint A: A foothold that causes legs crossing is

not allowed. This constraint improves the avoidance from deadlocks and leg collisions.

Constraint B: The L_{ML} foothold must satisfy a non-empty MSA_{NML}

Constraint C: The L_{NML} foothold must satisfy a non-empty CSW_{ML} for all next possible steps (ML=1,2,3,4).

Constraint B assure that a possible sequel step exist while constraint C increases the ability to move in every direction and also increases the overall stability by not allowing the legs to gather together in a small area. The downside of those constraints is the reduction of current *LWA*.

The local minima problem is a known problem when using a potential function [3], in the suggested algorithm, the use of DFS graph searching can help solving this problem. By remembering all previous steps, it is possible to add another constraint as follows:

Constraint D: A L_{NML} foothold cannot create a L vector that has been chosen before.

Constraint D can be tuned by changing the forbidden similarity to previous L vectors. Applying constraint D can be visualized as filing a local minimum with consecutive steps. This solution for the local minima problem prevent situations in which the robot steps at the same place or entering a loop of same steps because these are the best possible steps. A deadlock will occur only if the DFS will eliminate all possible steps and will return to the initial posture (q_0), this can happen because it is impossible to complete the required path because of the terrain conditions. The implementation of the above four constraints is done by eliminating foothold locations which do not satisfy the constraints.

The output of the main stage is a set of footholds that will navigate the *BP* along the path with the specific motion characteristics. However, this set of footholds is not necessarily the optimal set by meaning of the shortest *BP* path. When local minima are encountered while searching the graph the result is a suboptimal set of a foothold sequence for a given terrain and a required path. The algorithm's Pseudo-programming is shown in Algorithm 1.

In order to improve the solution of the first stage of the algorithm, a post-processing stage is needed. The goal of this stage is to identify the local minima and to create a new path by forcing the original path to pass through intermediate points, those specific points are chosen based on the solution of the first algorithm and they are located right after each local minimum. Then we run the algorithm first stage again with the intermediate way-points as the input path.

The suggested two stages algorithm is designed to run off-line and not in real-time walking. Although it is possible to use it for real-time application when prior terrain information is either known or acquired using a perception system. In this mode the ground surface data will be perceived using the sensors and the *LWA* will be calculated in realtime based on sensors data. Obviously in realtime use, the algorithm post-processing stage will be disabled and the *BP* rout will match the DFS exploration route. This realtime mode works best when all the required input path is planned over a passable ground so the DFS route will include forward motion only and no local minima. A real-time remote control of the robot can also be possible, by scanning the close by ground surface and creating a virtual continually changing path and adjusting the APF weights it is possible to "drive" the robot as desired while keeping all the algorithm stabilization capabilities. Hence, turning in place can be done by setting the path end point to be at the current *BP*, the robot will make a turn toward the path orientation while keeping its current position, side walking can be preformed by zeroing w_1 and planning a path parallel to the robot current heading.

Algorithm 1: Algorithm first stage

input : Robot kinematics, APF weights, $f_{BP}(q)$, q_0, MSM

output: A sequence of navigable stable postures

while $U > LPT$ **do**
  Calculate CKW;
  foreach *Possible two steps sequence (j=1...8)* **do**
    Calculate $FER \xrightarrow{MSM} MSA$;
    $CSW_{ML_j} = CKW \cap MSA_{ML_j}$;
    foreach COG_{jk} *in* CSW_{ML_j} *(k=1,2...)* **do**
      $LWV_{ML_{jk}} = f_{LWV}(COG_{jk} \in CSW_{ML_j})$;
      $LWA_{ML_{jk}} = LWV_{ML_{jk}} \cap Ground$;
      Apply constraints A,B and D over $LWA_{ML_{jk}}$;
      foreach $L_{ML_{jkn}}$ *in* $LWA_{ML_{jk}}$ *(n=1,2...)* **do**
        $LWV_{NML_{jkn}} = f_{LWV}(L_{ML_{jkn}} \in LWA_{ML_{jk}})$;
        $LWA_{NML_{jkn}} = LWV_{NML_{jkn}} \cap Ground$;
        Apply constraints A,C and D over $LWA_{NML_{jkn}}$;
        foreach $L_{NML_{jkni}}$ *in* $LWA_{NML_{jkn}}$ *(i=1,2...)* **do**
          $U_{jkni} = f(COG_{jk}, L_{ML_{jkn}}, L_{NML_{jkni}})$
  if *A possible step exist (i >1)* **then**
    $U_{min} = min(U_{jkni})$, for all j,k,n,i;
    Define current posture: $q = arg(U_{min})$;
    Log q;
  else
    DFS eliminate current q;
    Current q = previous q;
  if *current* $q = q_0$ **then** Deadlock;

IV. Simulation results

In this section we present the results of two simulation runs, the algorithm was implemented in MATLAB©[1]. The quadruped robot model is shown in figure 4, the legs *LWV* was modeled as cones therefore the *LWA* over a planner ground will be a disc while the body is parallel to the ground plane. We used this modeling as an approximation of the actual robot we are planning to conduct future experiments with (Figure 7). We chose straight lines to be the desired pathes.

[1]MATLAB is a registered trademark of MathWorks.

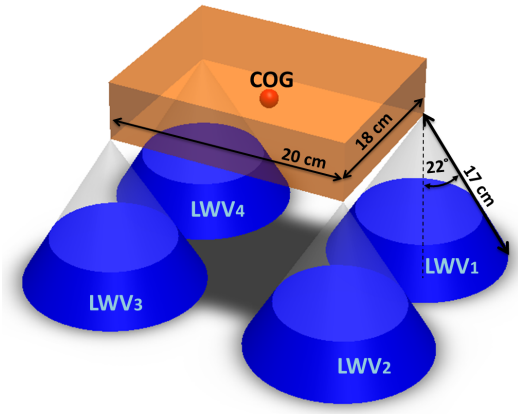
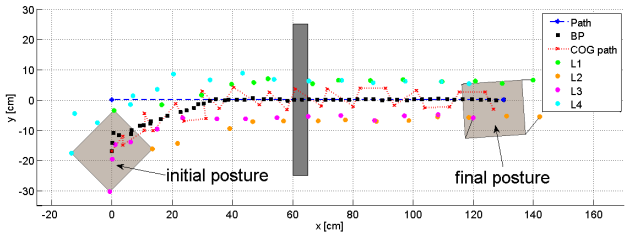
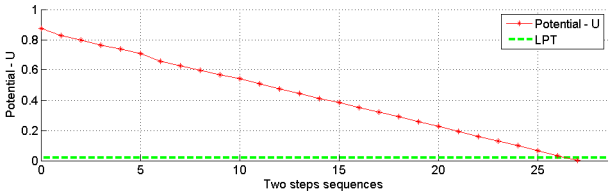


Fig. 4. Quadruped robot model

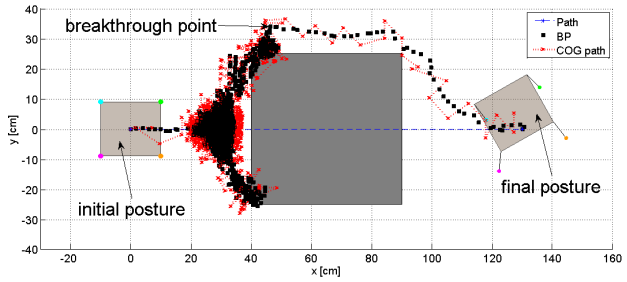


(a) The algorithm footholds output

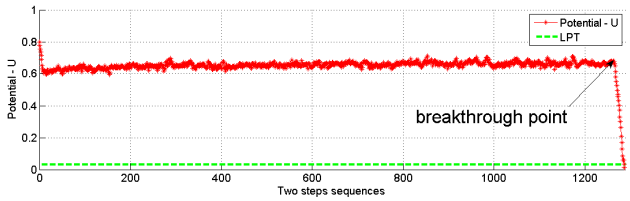


(b) Potential function change

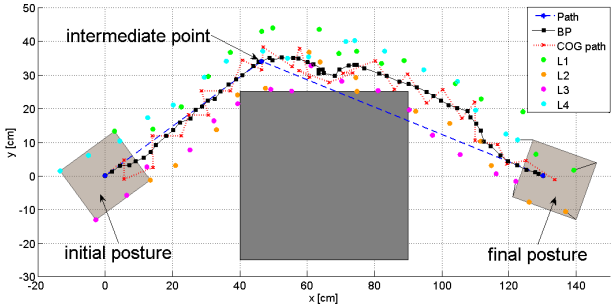
Fig. 5. Simulation of converging to a desired path and passing over a deep ditch.



(a) Algorithm footholds output



(b) Potential function change



(c) The algorithm footholds output after post-processing

Fig. 6. Simulation of the robot going around a large deep ditch.

In order to simplify the calculation complexity we reduce the *CKW* to two dimensions instead of three by defining the ϕ parameter as a function of the C_x and C_y parameters. We choose a function that will give a minimum leg twist about the body yaw angle, in our implementation the minimum leg twist is the average angle between the line connecting the two supporting side legs and the line perpendicular to the line connecting the two supporting front or real legs. We implemented the minimization problem solution using discreization of the search space. The following values were used for the algorithm as the user defined parameters:
MSM=2.5[cm].
w_1=1, w_2=0.15, w_3=0.15.
LPT=0.02.
Constraint D identity threshold: 2[cm].
Discretization resolution: 50 discrete points per *LWA*.
Depth of search: 2 steps (one double step sequence).
With the above parameters, using an Intel i7 2.67 GHz computer, the calculation time for a two step sequence was about 4 seconds, in this time about 9700 possible footholds combinations was compared.

The first simulation goal is to demonstrate the contribution of each component of the potential function and to verify the stability model by planning the required path to go over a deep ditch. when stepping inside the ditch we get an empty *MSA* so footholds cannot be placed there, all other ground surface is flat. Figure 5(a) shows the algorithm first stage output, each leg footholds is described in a different color and we can see the *BP* and *COG* pathes. The effect of w_1 and w_2 is moving the BP forward and closer to the path and w_3 is turning the body toward the path direction. Figure 5(b) shows the descent of the potential function while moving along the path. In this figure the x-axis units is the number of steps sequences so each unit defines two legs transitions. This simulation result did not require the use of the post-processing stage, in fact, because of no local minima the output after applying the second stage will match the output of the first stage.

The second simulation goal is to verify the algorithm post-processing stage, all simulation conditions are the same as the first simulation except of the ditch which is now wider and the final *BP* path will have to go around it instead of passing over. Figure 6(a) shows the output of the first stage of the algorithm (without the footholds), we can see that the

DFS algorithm is trying to find a way around the obstacle by changing between postures while trying to minimize the potential. Only after covering and eliminating all postures before the obstacle (i.e. filling the local minimum) there is a breakthrough and the potential is starting to descend again. This posture changing until the breakthrough is the local minimum filling phase and it is characterized by even or slow increases of the potential as shown in figure 6(b). After applying the post-processing stage in order to create an alternative path we get a new path with an intermediate point located at the breakthrough point of the first algorithm stage, the results of the algorithm first stage with the new path is shown in figures 6(c). The simulations results are also presented in the multimedia (video) extension of this paper.

V. Conclusions and future work

This paper has presented a two stage algorithm for free gait generation for quadruped robots, the algorithm output can be used to stably navigate a robot with omnidirectional capabilities along a desired motion path over rough terrain. Simulation results was presented to demonstrate and verify the algorithm functionality.

Future work will be divided to two directions, one is future development of the algorithm by analysis of the algorithm complexity and improving its performance while implementing the FER calculation in order to handle extremely rough terrains. The second direction is toward a quadruped robot devolepment for future experiments, we are currently planning experiments with a small robot, Figure 7 while our full scaled robot, Figure 8, is still under development.

References

[1] Shaoping Bai, K.H. Low, G. Seet, and T. Zielinska. A new free gait generation for quadrupeds based on primary/secondary gait. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1371–1376, 1999.

[2] Tim Bretl, Sanjay Lall, Jean claude Latombe, and Stephen Rock. Multi-step motion planning for free-climbing robots. In *in WAFR*, pages 1–16, 2004.

[3] Howie Choset, Kevin M. Lynch, Seth Hutchinson, GeorgeA Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[4] Jeffrey Dean, Thomas Kindermann, Josef Schmitz, Michael Schumm, and Holk Cruse. Control of walking in the stick insect: From behavior and physiology to modeling. *Auton. Robots*, 7:271–288, November 1999.

[5] C. Eldershaw and M. Yim. Motion planning of legged vehicles in an unstructured environment. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3383–3389, 2001.

[6] Joaquin Estremera and Pablo Gonzalez de Santos. Generating continuous free crab gaits for quadruped robots on irregular terrain. *IEEE Transactions on Robotics*, 21(6):1067–1076, 2005.

[7] Joaquin Estremera and Pablo Gonzalez de Santos. Free gaits for quadruped robots over irregular terrain. *The International Journal of Robotics Research*, 21(2):115–130, 2002.

[8] Shigeo Hirose. A study of design and control of a quadruped walking vehicle. *The International Journal of Robotics Research*, 3(2):113–133, 1984.

[9] H. Igarashi and M. Kakikura. Path and posture planning for walking robots by artificial potential field method. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 2165–2170, may 2004.

[10] H. Igarashi, T. Machida, F. Harashima, and M. Kakikura. Free gait for quadruped robots with posture control. In *9th IEEE International Workshop on Advanced Motion Control*, pages 433–438, 2006.

[11] J.Z. Kolter, M.P. Rodgers, and A.Y. Ng. A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.*, pages 811–818, may 2008.

[12] P. Maes, R. Brooks, Pattie Maes, and Rodney A. Brooks. Learning to coordinate behaviors, 1990.

[13] Yizhar Or and Elon Rimon. Analytic characterization of a class of three-contact frictional equilibrium postures in three-dimensional gravitational environments. *The International Journal of Robotics Research*, 29:3–22, 2010.

[14] Daniel J. Pack and HoSeok Kang. Free gait control for a quadruped walking robot. *Laboratory Robotics and Automation*, 11(2):71–81, 1999.

[15] P.K. Pal and K. Jayarajan. Generation of free gait-a graph search approach. *IEEE Transactions on Robotics and Automation*, 7(3):299–305, jun 1991.

[16] P. Gonzalez De Santos and M. A. Jimenez. Path tracking with quadruped walking machines using discontinuous gaits. *Computers & Electrical Engineering*, 21:383–396, 1995.

[17] Amir Shapiro, Elon Rimon, and Shraga Shoval. A foothold selection algorithm for spider robot locomotion in planar tunnel environments. *The International Journal of Robotics Research*, 24(10):823–844, 2005.

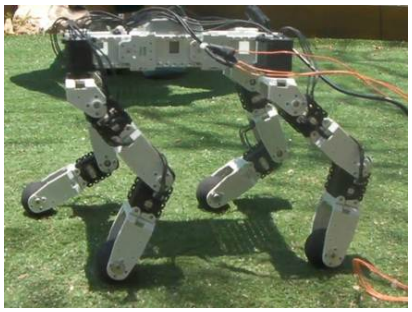


Fig. 7. Quadruped robot for future experiments

Fig. 8. Quadruped robot under development